Introduction to C Programming

This presentation will introduce you to the fundamentals of C programming. We will cover the history of the C language, its syntax, control structures, functions, arrays, pointers, and how to handle input and output. By the end of this session, you will have a solid understanding of the basics of C programming and be ready to dive deeper into more advanced topics.

CONTENTS(1)

CONTENTS(2)

Overview of C Language

History

Developed in the early 1970s by Dennis Ritchie.

Features

Structured, procedural, and low-level access to memory.

Applications

Used in system programming, embedded systems, and more.

Setting Up the Environment

1. **Install a Compiler**

   Choose a suitable C compiler for your system.

2. **Set Up an IDE**

   Install an Integrated Development Environment for easier coding.

3. **Configure Environment Variables**

   Ensure your system recognizes the compiler commands.

Setting Up the Environment

1. **Install Compiler**

   Install a C compiler (e.g., GCC, Clang).

2. **Choose IDE**

   Choose an Integrated Development Environment (IDE) or text editor (e.g., Code::Blocks, Visual Studio Code).

3. **Configure Variables**

   Configure the environment variables for easy access to the compiler.

4. **Write First Program**

   Write your first C program in a text editor.

Basic Syntax of C

Variables

Used to store data values.

Data Types

Defines the type of data a variable can hold.

Operators

Symbols that perform operations on variables and values.

Basic Syntax of C

Comments

Use // for single-line comments and /* ... */ for multi-line comments.

Data Types

Common types include int, float, char, and double.

Variables

Declare variables using the syntax data_type variable_name;.

Operators

Arithmetic, relational, and logical operators are used for operations.

Control Structures

1. **If Statements**

   Used for conditional execution.

2. **Loops**

   For repeating a block of code.

3. **Switch Cases**

   Used for multi-way branching.

Control Structures

Conditional Statements

if, else if, and else for decision-making.

Switch Statement

A multi-way branch statement.

Loops

for, while, and do-while loops for repeated execution.

Functions in C

1 Definition

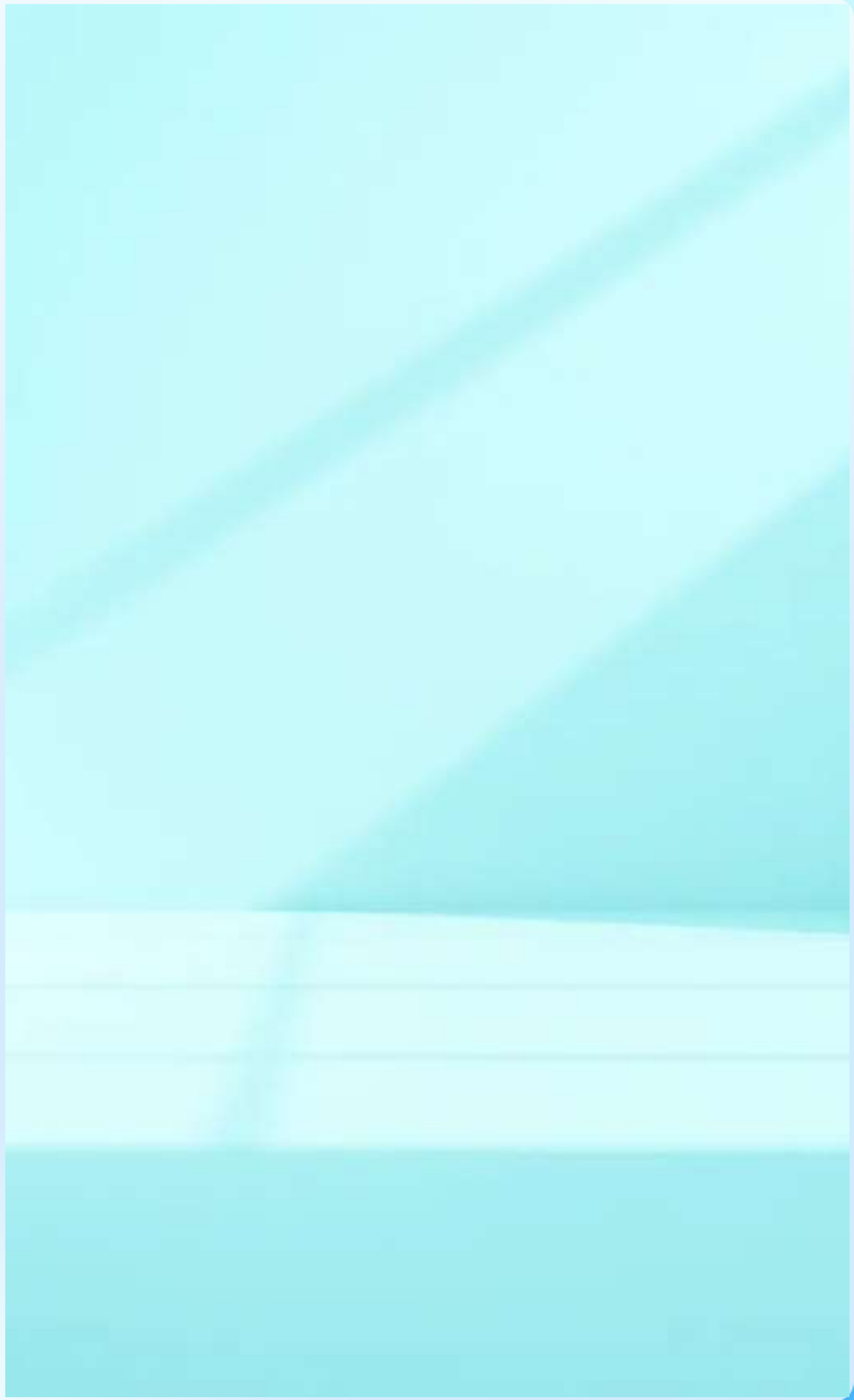
A function is a block of code that performs a specific task.

2 Syntax

Functions have a specific syntax including return type, name, and parameters.

3 Usage

Functions help in code reusability and organization.



Functions in C

Function Declaration

Syntax is return_type function_name(parameters).

Function Definition

Contains the code to be executed.

Function Call

Invoking a function using its name and passing arguments.

Scope

Variables defined inside a function are local to that function.

Arrays and Pointers

1 **Definition**

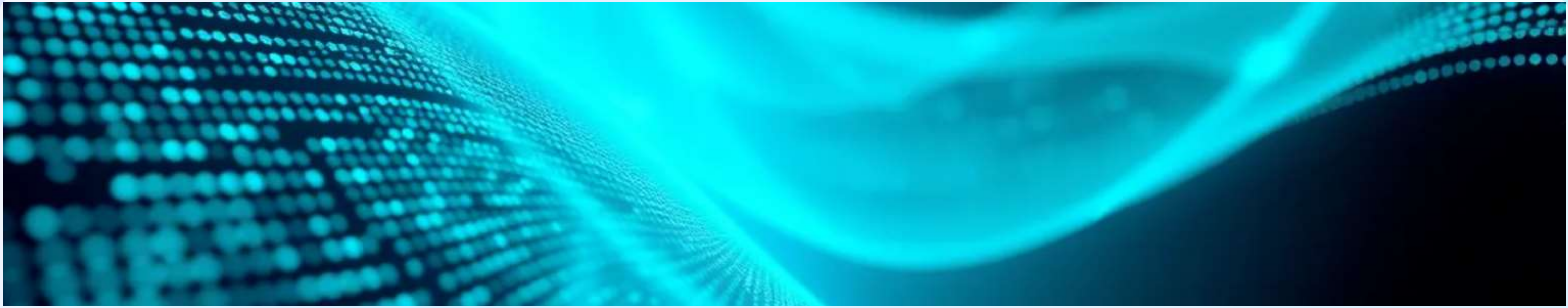
Arrays are collections of elements.

2 **Pointers**

Pointers store memory addresses.

3 **Usage**

Used for dynamic memory allocation.



Arrays and Pointers

Arrays

A collection of elements of the same type, declared as data_type array_name[size].

Pointers

Variables that store memory addresses, declared as data_type *pointer_name.

Pointer Arithmetic

Allows manipulation of array elements using pointers.

Input and Output

Standard Input

Reading data from the keyboard.

Standard Output

Displaying data on the screen.

File Input

Reading data from files.

File Output

Writing data to files.

Input and Output

1 **printf**

Used for outputting data to the console.

2 **scanf**

Used for reading input from the user.

3 **File I/O**

Functions like fopen, fclose, fprintf, and fscanf for file operations.

Conclusion and Next Steps

Review Key Concepts

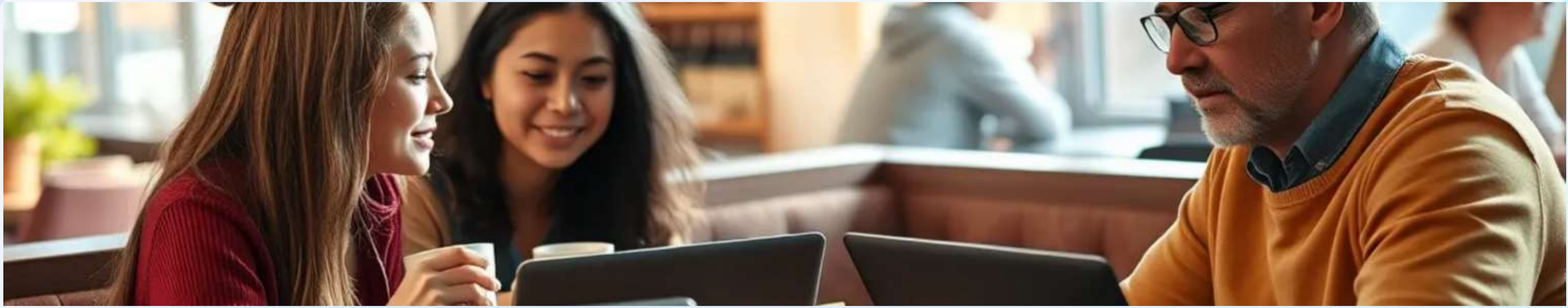
Revisit the main topics covered in the course.

Practice Coding

Engage in hands-on coding exercises to reinforce learning.

Explore Advanced Topics

Consider diving into more complex areas of C programming.



Conclusion and Next Steps

1. **Practice Programming**

   Practice writing simple programs.

2. **Explore Advanced Topics**

   Explore advanced topics like structures and dynamic memory allocation.

3. **Engage with Communities**

   Engage with online coding communities for support and resources.

Thank You

1 Appreciation

Thank you for your attention.

2 Questions

Feel free to ask any questions.

3 Next Steps

Looking forward to your feedback.